REMARKS

Claims 1-27 are now pending. Applicant has amended claims 2, 8, and 12 to correct some minor wording concerns. Applicant has added claims 16-27 to more comprehensively claim the invention.

Applicant has added the legend "Prior Art" to Figure 3 as requested by the Examiner.

Applicant has corrected a minor typographical error in the specification that was pointed out by the Examiner.

The Examiner has objected to claim 5 under 37 C.F.R. § 1.75(c) as being of improper dependent form for failing to limit the subject matter of a previous claim. Applicant respectfully disagrees. Applicant is entitled to claim the invention using the language of applicant's choice. In this case, the term "invoking" further limits the term "exercising," because "invoking" as a term of art in computer science may be considered more precise than the term "exercising."

The Examiner has rejected claims 2, 8, and 12 under U.S.C. § 112, second paragraph, as being indefinite. Although applicant disagrees, applicant has amended the claims to address the Examiner's concerns.

The Examiner has rejected claims 1-10 under 35 U.S.C. § 102(e) as being anticipated by Kobayashi, claim 11 under 35 U.S.C. § 103(a) as being unpatentable by Kobayashi in view of Rodrigues, and claims 12-15 under 35 U.S.C. § 103(a) as being unpatentable over Kobayashi in view of Rodrigues and MTS. Applicant respectfully disagrees.

The Examiner misunderstands Kobayashi. Kobayashi describes a typical visual programming environment. Kobayashi automatically generates a proxy component for each method of an underlying component. A proxy component provides the interface

between the visual programming environment and its method in the underlying component. A proxy component has properties that correspond to the actual parameters that are supplied when the method of the underlying component is invoked. A programmer creates the program by setting the properties of the proxy components, which are passed as actual parameters when the method of the underlying component is invoked.

Referring to Figure 11 of Kobayashi, a programmer can create a composite component by dragging proxy components from the bean palette window 1125 to the workspace panel window 1140. The programmer can set the values of the properties of the proxy component using the properties panel 1155 that are to be used when the method of the underlying component is invoked. The programmer can then wire the proxy components together by event handling or by binding properties. (Kobayashi, 19:2-6.) To wire proxy components by event handling, a programmer selects a source proxy component and a target proxy component and specifies an event type. The source and target proxy components are wired in the sense that when the source proxy component generates that event type, the target proxy component is executed, resulting in the invocation of the method of the underlying component. To wire proxy components by binding properties, a programmer selects a source property of a source proxy component and a target property of a target proxy component. The source and target proxy components are wired in the sense that when the source property is changed that change is propagated to the target property.

Once a programmer has created a composite component, the programmer can test it. Kobayashi describes the testing with reference to Figure 20. In step 2004, the programmer loads the composite component into the workspace window. In step 2006, the programmer executes the composite component by selecting a "Run" button of the visual programming environment. In step 2008, the component is executed. In step 2010, the programmer checks to see if the composite component executed properly. The programmer can check to see if images were displayed correctly, if bound properties changed correctly, and so on. If the composite component did not execute correctly, then

the programmer in step 2014 can attempt to correct the problem (i.e., find and fix the bug) and repeat the testing.

Applicant's claims are directed to a technique for testing an object. Applicant's technique allows a programmer to select an object, a method of the object, and parameters for the method. The technique then invokes the selected method of the selected object using the selected parameters. The programmer can then check the result and continue testing the object by selecting a new method and/or new parameters. Each of the claims recites this technique. Claims 1-5 recite "retrieving from an input dialog a selected method" and "obtaining selected parameters for use in the method for exercising the selected object." Claims 6-11 recite "receiving input dialog selections that specify a selected object and a selected method" and "displaying an input dialog for parameters to provide selected parameters for use by the selected method." Claims 12-15 recite "detecting an input selection indicating each object to be exercised and the method to be exercised" and "getting the method parameters chosen for use with the method to exercise the instance of the object."

The Examiner is relying Kobayashi's testing loop of Figure 20 in rejecting these claims. Kobayashi's testing tests a composite bean to ensure that it is correctly programmed in a manner similar to how a programmer would test a program in a non-visual programming environment. Each instance of a proxy component corresponds to an invocation of a method of the underlying component using the properties of the proxy component as actual parameters. This is analogous to a "call" statement in a conventional programming environment. Thus, the proxy components that make up a composite component are in effect the statements of the program. Kobayashi's testing is equivalent to executing a conventional program during testing. The program is executed and the statements are changed in an attempt to fix problems.

Kobayashi does not teach or suggest "retrieving from an input dialog a selected method." The Examiner believes that Kobayashi's selecting of a composite component (or

bean) in step 2004 corresponds to the claimed retrieving. This selecting of Kobayashi does not, however, select a method of an object as recited by the claims. Rather, Kobayashi's selecting selects a composite component itself. Kobayashi neither teaches nor suggests selecting any methods of that composite object from an "input dialog."

Moreover, Kobayashi does not teach or suggest selecting of parameters for the selected method. The only parameters can be found in Kobayashi's proxy components, which are hard-coded into the proxy components during visual programming. These parameters are not input from an input dialog when the composite component is being tested.

Claims 16-21 recite "exercising the instantiated object by repeatedly: displaying to a user a list of methods of the objects . . . and . . . invoking the selected method of the instantiated object passing the selected parameters." Claims 22-27 recite "providing entries that specify an object, a method of the object, and a parameter of the object; . . . and . . . for each entry, . . . invoking the method of the entry of the instantiated object with the parameter of the entry." Neither Kobayashi nor any of the other references teach such repeated invocation.

Based on the above amendments and remarks, applicants respectfully request reconsideration of this application and its early allowance. If the Examiner has any questions or believes a telephone conference would expedite prosecution of this application, the Examiner is encouraged to call the undersigned at (206) 359-8548.

Dated: 9/14/04

Respectfully submitted,

By Maurice J. Pirio

Maurice J. Pirio
Registration No.: 33,273
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorneys for Applicant